Effectiveness of Feature Selection and Machine Learning Techniques for Software Effort Estimation

Jyoti Shivhare

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela-769 008, Odisha, India
June 2014

Effectiveness of Feature Selection and Machine Learning Techniques for Software Effort Estimation

Thesis submitted in partial fulfillment of the requirements for the degree of

Master of Technology

in

Computer Science and Engineering

(Specialization: Software Engineering)

by

Jyoti Shivhare

(Roll No.- 211CS3126)

under the supervision of

Prof. S. K. Rath

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela, Odisha, 769 008, India

June 2014

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela-769 008, Odisha, India.

Certificate

This is to certify that the work presented in the thesis entitled ***Effectiveness of Feature Selection and Machine Learning Techniques for Software Effort Estimation*** by ***Jyoti Shivhare*** is a record of an original research work carried out by her under my supervision and guidance in partial fulfillment of the requirements for the award of the degree of Master of Technology with the specialization of Software Engineering in the department of Computer Science and Engineering, National Institute of Technology Rourkela. Neither this thesis nor any part of it has been submitted for any degree or academic award elsewhere.

Place: NIT Rourkela
Date: June 1, 2014

(Prof. Santanu Ku. Rath)
Professor, CSE Department
NIT Rourkela, Odisha

Acknowledgment

A student work is incomplete until she thanks the Almighty and her teachers. I sincerely believe in this and thank God for showing me the right direction.

I am thankful to many nearby and global associates who have helped towards modeling this thesis. I would like to convey my deepest gratitude to my supervisor Prof. Santanu Ku. Rath, for his excellent guidance, caring and providing me with an excellent atmosphere for doing research. His keen interest, patient hearing and constructive criticism have instilled in me the spirit of confidence to successfully complete this thesis. I am greatly indebted for his help throughout the thesis work.

Besides him, I am also appreciative to all the Professor and faculty members of the computer science department for their in time assistance, advice and encouragement.

I am really grateful to all my friends and lab-mates for their co-operation. My sincere thanks to Mr. Suresh, Mr. Mukesh, Sumana, Prerna, Amar Nath, and Lov for their support and help. I am truly indebted.

I do give thanks to all academic assets that I have been gained from NIT Rourkela. I would like to thank the managerial and specialized staff parts of the Computer Science Department for their in-time support.

Last, yet not the least, I might want to devote this thesis to my family for their steady help, patience, support, love, understanding, encouragement and co-operation.

Jyoti Shivhare

Roll-212CS3126

Abstract

Estimation of desired effort is one of the most important activities in software project management. This work presents an approach for estimation based upon various feature selection and machine learning techniques for non-quantitative data and is carried out in two phases. The first phase concentrates on selection of optimal feature set of high dimensional data, related to projects undertaken in past. A quantitative analysis using Rough Set Theory and Information Gain is performed for feature selection. The second phase estimates the effort based on the optimal feature set obtained from first phase. The estimation is carried out differently by applying various Artificial Neural Networks and Classification techniques separately. The feature selection process in the first phase considers public domain data (USP05). The effectiveness of the proposed approach is evaluated based on the parameters such as Mean Magnitude of Relative Error (MMRE), Root Mean Square Error (RMSE), Mean Absolute Error (MAE) and Confusion Matrix. Machine learning methods, such as Feed Forward neural network, Radial Basis Function network, Functional Link neural network, Levenberg Marquadt neural network, Naive Bayes Classifier, Classification and Regression Tree and Support Vector classification, in combination of various feature selection techniques are compared with each other in order to find an optimal pair. It is observed that Functional Link neural network achieves better results among other neural networks and Naive Bayes classifier performs better for estimation when compared with other classification techniques.

Keywords: Artificial Neural Network, Effort Estimation, Feature Selection technique, Machine Learning technique, Rough Set Analysis.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

An essential preliminary step while developing any software project is to have an idea on effort required to develop a software system. Effort is defined as the person months required to make a software application. A number of methods have been suggested to estimate the effort required for any project, as available in literature. These estimation methods can be categorised as: algorithmic and non-algorithmic. The commonly used algorithmic estimation models are Boehm′s COCOMO [1], Putnam′s SLIM and Albrecht′s Function Point [2]. The need for accurate cost estimation has led to the exploration of the non-algorithmic models which are based on machine learning techniques. These methods make minimal assumptions about the form of the function for development effort of project under study. These methods depend on historical data and construct the systems that can learn from data [3].

The work presented in this thesis has been carried out in two phases. The first phase discusses various feature selection techniques and the second phase is about implementation of machine learning techniques for software effort estimation. The approach given in this thesis focuses on the results of the application of machine learning models in the field of software effort estimation. The input to this model is the dataset obtained after performing feature selection in original dataset [4], using Rough Set Analysis (RSA) and Information Gain as these techniques emphasizes the role of feature selection in estimation method [5]. For the estimation of the effort for a project, various machine learning models have been designed. Four different Artificial Neural Network models are used. The data driven and self-

adaptive nature of the neural network model encourages one to consider it as an estimator. The network adjusts itself to the data without any explicit specification of the underlying model [6] [7]. The models used in this study are Feed Forward, Radial Basis, Functional Link and Levenberg Marquadt neural networks. Three different classifier named Naive Bayes classifier, Classification and regression tree and support vector machine are considered for effort estimation. The results are compared in order to find a pair of feature selection and machine learning technique which works better than those of other combinations.

1.1 Problem Statement

Accurate estimation of the effort needed for a software project is very important in software development life cycle. To deliver the project within the desired time and cost is necessary. In literature many models have been developed to solve this problem. The work presented in this thesis tries to resolve this issue to some extent.

1.2 Motivation

Estimating development effort is primal to the management and control of software project. If the estimated effort is too less, then the developers may have pressure to finish the product quickly and hence the resulting software may not be fully functional or tested. On the other hand, if the estimated value is too high, then too many resources will be engaged in the project, resulting in poor resource utilization. To help the industry in developing quality products within scheduled time, accurate software effort estimation is necessary.

1.3 Literature Review

A number of methods have been suggested in literature till date to estimate software development effort. Among these, the initial models are algorithmic in nature, which estimate the effort needed to develop a software using a formula being

generalized from historical data.

Berry Bohem had developed the Constructive Cost Model (COCOMO) [1] based on a linear regression analysis of sixty three projects. Bohem, in his COCOMO model relates the effort needed to develop a software project to Delivered Source Instructions (DSI). Putnam [8] has given a model, Software LIfe cycle Management, called as SLIM, which estimates the effort of software project by using SLOC (Source lines of code) as the major input. Albrecht [2] developed Function Point method based on features of the software project that are at a higher descriptive level than SLOC, for example the number of internal data file, external data file and number of reports etc.

More recently, a large number of machine learning methods originating from the data mining literature are being used for effort estimation. These include several regression methods in a linear model and nonlinear techniques like neural networks and rule based models. A brief summary of literature work for software effort estimation is given next.

In the work of Krishnamoorthy Srinivasan et al. (1995) [3] different Machine learning techniques are used for estimating development effort. These techniques include Learning Decision and Regression Trees and Artificial neural network. The results are compared with traditional approaches like COCOMO and SLIM model. Martin Shepperd et al. (1997) [9] have first characterized the past projects in terms of attributes and then similarity technique is used to estimate the effort for a new software project. Euclidean distance method is used to find most similar projects. Barry Boehm et al. (2000) [10] have done a survey of different software development cost estimation approaches existing in literature and came to a conclusion that no single technique exist which is best for all situation.

In the work of Zhihao Chen et al. (2005) [11], the authors explain that COCOMO's estimates can be improved with the use of feature subset selection method, WRAPPER. They have used dataset from PROMISE repository and conclude that WRAPPER significantly improves predictive power of COCOMO. Parag C Pendharkar et al. (2005) [12] used a Bayesian probabilistic model and

illustrates an procedure that can be used in decision making risks. The efficiency of the proposed model is compared with artificial neural network and regression tree prediction models, with conclusion that probabilistic analysis gives better results. In the study of Adriano L.I. Oliveira (2006) [13], the estimation of software project effort is carried out with three different techniques namely support vector regression (SVR), radial basis function neural network (RBFN) and linear regression. The study concluded that SVR significantly outperforms the other two technique. Jingzhou Li et al. (2007) [14] proposed an approach which supports non-quantitative attributes and tolerates missing values in dataset. They have performed effort estimation for different kinds of projects using case based reasoning. Next in (2008) [15], they present a work which predicts the effort for a new software project by aggregating the effort information of similar past projects. They have used Rough Set Analysis as a pre-step to perform weighting and selection of attributes. Effort is estimated using similarity measure techniques.

B. Tirimula Rao et al. (2009) [16] have proposed an approach for estimating the cost of software project using Functional Link Artificial Neural Network (FLANN). The approach initially uses COCOMO method to predict the cost of software and then uses FLANN technology with backward propogation. In the study by Yan-Fu Li et al. (2009) [17], effort is estimated using case based reasoning. Analogy based estimation model is incorporated with Artificial neural network and the proposed model is able to handle categorical data. Prasad Reddy et al. (2010) [18] have used two types of Artificial neural network, Radial Basis networks and Generalized Regression neural network, with COCOMO dataset in order to find the neural network which performs better with COCOMO model for software effort estimation. In 2012, Karel Dejaeger et al. [19] have done a comparative study of different linear and non-linear models, already existing in literature, in order to find the best technique. The model includes CART, Multilayered Perceptron Neural networks, MARS, Case-based Reasoning approach etc. Ekrem Kocaguneli et al. (2012) [20] have proposed an Analogy based estimation (ABE) model with non-uniform weighting of attributes through kernel density functions. They come

to a conclusion that simple ABE approach is able to perform better than complex proposed approach.

Hence it is observed that there have been a variety of models developed for estimating development effort. They assume that an initial estimate can be framed on empirical basis, which can fit into historical data.

1.4 Various Performance Measures

Different performance measures for evaluating the efficiency of estimation model have been proposed in the literature [21]. Among those the following criteria are considered to evaluate the performance of machine learning methods for software effort estimation.

1.4.1 Performance measures for Artificial neural network models

Artificial neural network is a machine learning model, which is used in this study as a predictor. The performance of ANN model is evaluated using different measures explained next.

- Root Mean Square Error (RMSE):

  RMSE computes the difference between value estimated by a model and the value actually observed. It is the square root of the mean square error, as given in equation:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(X_i - Y_i)^2} \tag{1.1}$$

- Mean Magnitude of Relative Error (MMRE):

  MMRE measures the difference between estimated and actual value relative to actual value, as given in equation:

$$MMRE = \frac{1}{N}\sum_{i=1}^{N}\frac{|X_i - Y_i|}{X_i} \tag{1.2}$$

- Mean Absolute Error (MAE):

  The mean absolute error is a measure of how far the estimates are from

actual values. MAE is defined as:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|X_i - Y_i| \tag{1.3}$$

where

X_i: Actual value of data point i

Y_i: Estimated value of data point i

$|X_i - Y_i|$: Absolute value of $X_i - Y_i$ and

N: Total number of data points.

1.4.2 Performance measures for Classification models

The performance parameters for statistical analysis of different classification models are defined based on the *confusion matrix* as shown in Table 1.1.

Table 1.1: A Confusion Matrix

		Predicted	
		Yes	**No**
Actual	**Yes**	True Positive (TP)	False Negative (FN)
	No	False Positive (FP)	True Negative (TN)

The *confusion matrix* has four cells for a two class prediction problem, which are defined next:

i. True positives (TP): is the number of objects, which are correctly classified. It shows that actually object belong to class *Yes* and classifier result is also class *Yes*.

ii. False positives (FP): refer to the number of objects, which are actually member of class *No* but classified as class *Yes*.

iii. True negatives (TN): is the number of objects of class *No*, which are classified as *No* by classifier.

iv. False negatives (FN): refer to the number of objects, which are actually member of class *Yes* but classified as class *No*.

The following are the performance measures used in classification.

- Precision

  Precision can be explained as the fraction of total positive cases that were correctly classified to the total cases classified as positive. The Equation 1.4 calculates the Precision as:

$$Precision = \frac{TP}{TP + FP} \tag{1.4}$$

- Recall

  The term Recall is best known as sensitivity or true positive rate (TPR). It is expressed as the proportion of the positive cases that were correctly classified to the total actual positive cases.

$$Recall = \frac{TP}{TP + FN} \tag{1.5}$$

- Accuracy

  The term Accuracy can be defined as the overall correctness of the classification model. It is calculated as the ratio of the sum of correct classifications to the total number of classifications.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{1.6}$$

1.5 Dataset used for Software Effort Estimation

Two public domain data sets USP05-FT and USP05-RQ with non-quantitative attributes from PROMISE Software Engineering Repository are considered in this study to perform effort estimation [4]. USP05 (University Student Projects developed in 2005) was collected from projects developed by students about web or client/server applications. USP05-FT and USP05-RQ has projects data at feature level and requirement level respectively. The projects are characterized using 14 attributes, which are continuous, discrete and categorical in nature. The detailed definition of attributes is presented in Table 1.2. Both datasets have same attributes but the domain is different for categorical attributes. The range of

discrete attribute is also different. The existing work done using this data set is given in [22] [14].

The data has attributes, for which the value can be identified after the first phase, i.e. Requirement Gathering & Analysis phase, of software development life cycle. So a new project can be defined using these attributes. Henceforth the proposed approach is applicable to estimate the effort for a new software project after the requirement analysis phase.

Table 1.2: Description of attributes of data set

Name	**Description**
ID	Object ID
Effort	This attribute gives the number of hours expended on a tasks needed to complete the project by all team members.
IntComplx	The value of this attribute shows the complexity level of internal calculation.
DataFile	It gives the number of data files and database tables created and accessed by project.
DataEn	The total number of data input items.
DataOut	The total number of data output items.
UFP	This feature gives the final value of unadjusted function point count.
Lang	The language used to develop the project is presented by this attribute.
Tools	The tools and platforms used for a particular project are given by this attribute.
ToolExpr	This attribute gives the value of language and tool experience level of project team, for e.g., $[a, b]$ for a to b months, as the minimum tool experience required is a months and the maximum required level is b months in the team.
AppExpr	The level of experience for a particular application or in other words the experience level of project team about the application domain.
TeamSize	The Team size for developing project. for e.g. [a, b] where a is the lowest and b is the highest number of persons who are part of the development team of the project.
DBMS	The database management systems used for implementing the project.
Method	The methodology used for building the project.
AppType	The type of System or Application architecture is given in this field where B is for Browser, C for Client and S is for Server.

1.6 Thesis Organization

The work done in this thesis has been organized in following way:

- **Chapter 2: Feature Selection**: Different feature selection techniques are given in this chapter. Rough set analysis and information gain methods are discussed and implemented. The results of these techniques are given which form the basis for the next Chapter.

- **Chapter 3: Software Effort Estimation using Machine Learning Techniques**: In this chapter, different Machine learning techniques like Artificial neural network, Naive bayes classifier, Decision tree and Support vector classifier have been suggested and implemented for software effort estimation. The results of these models are compared to find a suitable feature selection and machine learning technique.

- **Chapter 4: Conclusion and Future Work**: Finally this chapter concludes the work done in this study with scope of future work.

Chapter 2

Feature Selection

2.1 Introduction

The efficiency of machine learning methods depends on the quality of data used to train the model. The data can be irrelevant, incomplete, redundant and noisy. For estimating the effort, all the features are not important, only a small subset of feature is relevant. So, selecting an optimal subset of features is a basic requirement of the proposed work. Feature subset selection is a procedure of first finding the relevancy of each feature, then identification and selection of the most relevant features. The selected features show high predictive power and can avoid the over fitting of the training data.

The existing literature shows that algorithms run faster and take less space when using the reduced data. The results are also improved for classification and can be easily understood. A large number of feature selection techniques exist in literature and can be divided into two categories. One which evaluates individual attribute and other which evaluates subsets of attributes [23]. The work in this study has been carried out using two different feature selection techniques, each belongs to one category. Rough Set Analysis is used in two different ways to perform feature selection. Another technique is Information Gain, which selects the features based on ranking.

2.2 Rough Set Analysis for Feature Selection

Rough set theory has been introduced by Zdzislaw Pawlak in 1982′s [24]. An important property of Rough set analysis (RSA) is that there is no requirement of any preliminary information about dataset. A concise description of rough set analysis is presented here. The dataset is mentioned to as *information system*

$$IS = (U, A)$$

where,
U is a non-empty finite set of *objects* called the *universe*, $U = \{x_1, x_2, ..., x_m\}$ and A is a non-empty finite set of *attributes* called *space*, $A = \{a_1, a_2, ..., a_n\}$.

Therefore every $B \subset A$ gives a binary relation on U which will be known as an *indiscernibility relation*, presented by $Ind(B)$. A attribute a_i is independent in A,

$$\text{if } Ind(A) = Ind(A - a_i)$$

and a set B is named as *independent* if all of its members are independent otherwise set is *dependent*.

An important property of rough set is that the decreased set of attributes provides the same level of information as the actual set of attributes. The reduced set of attributes of the information system, known as *reduct*, is independent and no other attribute can be further removed from the data without some loss of information. In other words, the *reduct* can be expressed as a minimum subset of attributes which offers the same classification of the objects of universe as the original set of attributes.

Any subset B of A is called a reduct of A, if B is independent and $Ind(B) = Ind(A)$. There are many ways to find *reduct* of an information system and more than one *reduct* are possible for an information system. In this work, *Dicernability Matrix* method has been used to find out *reduct* of dataset [25]. *Reducts* derived using rough set analysis are given as input to the machine learning models.

2.3 RSA for Feature Ranking and Selection

Another application of rough set analysis is to evaluate the feature at individual level. The result of Dicernability Matrix technique in Section 2.2 gives more than one reduct and perform feature selection at subset level. As there are more than one reduct, so it is possible that some features are part of more than one reduct. So in order to find out the relevancy at feature level, all the *reducts* found in Section 2.2 are considered.

The proposed idea assumes that the more frequent an attribute a_i appears in *Reduct(IS)*, the more the target attribute Effort is dependent on a_i. Suppose the number of occurrence of an attribute a_i in *Reduct(IS)* is N_i, then the weight of a_i is defined as:

$$w(a_i) = \frac{N_i}{\sum_{j=1}^{m} N_j} \tag{2.1}$$

where m is the total number of *reduct* of dataset. After calculating the weight of each attribute, only those attributes are considered in study which satisfy a threshold value.

2.4 Information Gain for Feature Ranking and Selection

Information gain is one of the easiest and quickest attribute ranking methods. It is widely used in the applications in which the dimensionality of data set is very high, for example text categorization. The prior uncertainty and the expected posterior uncertainty for a condition attribute A and class attribute C is calculated using Equation 2.2 and Equation 2.3.

$$H(C) = -\sum_{c \in C} p(c) log_2 p(c) \tag{2.2}$$

$$H(C|A) = -\sum_{a \in A} p(a) \sum_{c \in C} P(c|a) log_2 p(c|a) \tag{2.3}$$

The information gain from an attribute A is expressed as the difference between the prior and posterior uncertainty using A. A score is assigned for each attribute A_i on the basis of the information gain between A_i and class C using Equation 2.4.

$$InfoGain_i = H(C) - H(C|A_i) \tag{2.4}$$

Attribute A_i is preferred to attribute A_j if the InfoGain from A_i is greater than InfoGain from attribute A_j. After calculating the InfoGain for each attribute, only those attributes are selected which satisfy a threshold value [23].

2.5 Proposed Approach

To perform feature selection on data, the following steps has been carried out. Figure 2.1 shows the framework of feature selection model where the output is reduced data.

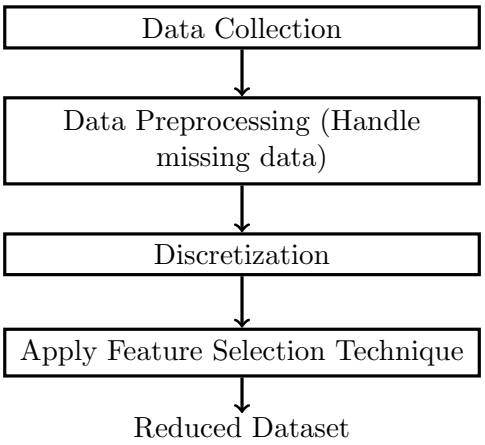

Figure 2.1: Framework of Feature Selection model

Steps in Feature Selection

1. **Data Collection**: This step includes the identification and collection of data from previously developed projects. This first stage is very much dependent upon the nature of the projects for which estimates are required.

2. **Data Preprocessing**: Data preprocessing is necessary to handle noisy data. All rows having missing data are omitted from dataset.

3. **Discretization**: All the continuous and the categorical values in data are converted into discrete values. Discrete data are used as input to feature selection model.

4. **Application of Feature Selection Technique**: Different feature selection techniques are applied on discrete data. The output of this step is the reduced dataset.

2.6 Implementation and Results

2.6.1 Pre-processing of Data

The data set used in the proposed work contains many missing values. So, rows containing missing values are omitted. The two data sets now have 57 and 102 objects respectively.

2.6.2 Discretization

Feature selection techniques considered in this work can only address discrete data in order to get reduced data. Thus, the continuous data must be discretized before applying any feature selection model. Discretization is performed by partitioning the continuous domain of the attribute into subintervals [26]. Two techniques, *Equal Frequency Interval* and *Equal Width Interval* are considered to discretize the continuous attributes of data in this work.

- *Equal Frequency Interval*: In this method m interval has been made such that each interval contains same number of instances of values.

- *Equal Width Interval*: This method divides the range of continuous value in equal sized intervals.

Since some attributes are already discrete, their value is represented by a single number. Table 2.1 shows the range of discretization for continuous attribute. A sample of discretized data is presented in Table 2.2.

Table 2.1: Discretized attribute

Effort	DataFile	DataEn	UFP	ToolExpr	TeamSize
1 (Low)—[0,1]	1—[0,1]	1—[0,1]	1—[0,2]	1— $[0, 12]$	1— $[1, 1]$
2 (Fair)—(1,3]	2—(1,3]	2—(1,3]	2—(2,5]	2— $[0, 60]$	2— $[1, 2]$
3 (Medium)—(3,6]	3—(3,5]	3—(3,7]	3—(5,13]	3— $[1, 10]$	3— $[2, 3]$
4 (High)—(6,25]	4—(5,9]	4—(7,90]	4—(13,64]	4— $[2, 60]$	4— $[1, 3]$
				5— $[4, 24]$	
				6— $[5, 100]$	

Table 2.2: Sample of USP05-FT data set after performing Discretization

ID	Effort	Int Com-plx	Data File	Data En	Data Out	UFP	Lang	Tools	Tool Expr	App Expr	Team Size	DBMS	Method	SA Type
114	2	2	4	4	0	1	HPSP	NW	4	4	1	MySQL	SASD	BCS
116	3	2	4	4	0	1	HPSP	NW	4	5	1	MySQL	SASD	BCS
119	2	2	4	4	0	1	HPSP	NW	4	5	1	MySQL	SASD	BCS
206	1	1	2	3	1	1	PHSJ	VEM	3	1	3	Oracle	3Tier	BCS
211	2	1	2	1	1	1	PHSJ	VEM	3	2	2	Oracle	3Tier	BCS
220	1	1	3	3	1	1	PHSJ	VEM	3	2	2	Oracle	3Tier	BCS
224	1	1	2	2	1	1	PHSJ	VEM	3	2	2	Oracle	Oracle	BCS
402	4	2	4	2	2	4	PS	Pico	6	2	1	MySQL	OO	BS
403	2	3	2	2	1	3	PS	DREP	5	4	1	MySQL	OO	BS
622	4	2	2	2	1	3	PMH	Dream	2	1	1	MySQL	Imperative	BCS
715	1	1	2	2	0	2	PHP	Notepad	1	4	2	Oracle	OO	BS
718	1	1	2	1	2	3	PHP	Notepad	1	4	2	Oracle	OO	BS
801	4	3	1	4	1	4	PHP	Emacs	1	4	3	MySQL	OO	BCS
804	3	2	1	4	1	4	PHP	Emacs	1	4	2	MySQL	OO	BCS
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

2.6.3 Feature Selection using RSA

Attribute selection technique is required to apply in order to reduce the number of attributes in data. As all the attributes present in the collected data are not relevant to estimate the target attribute, number of attributes are reduced by performing RSA on data. *Discernibility matrix* technique of RSA is used in this work to perform feature selection. This technique first find out all possible subset of attribute set and then take out only those subsets which are independent.

The result of RSA is shown in Table 2.3. Initially the project′s data collected has fifteen features and after applying RSA on data, number of attributes for USP05-FT data is reduced to six including effort. From Table 2.3 it is evident that more than one reduct of dataset are existing and in the next phase of approach, the classifiers can use any of these reduct. The size of reduct for USP05-RQ dataset is seven.

Table 2.3: Sample of Reducts for USP05-FT data

Reduct No.	**Attr1**	**Attr2**	**Attr3**	**Attr4**	**Attr5**
1	IntComplx	DataEn	AppExpr	TeamSize	SAType
2	IntComplx	DataEn	AppExpr	TeamSize	Method
3	DataFile	DataEn	AppExpr	TeamSize	SAType
4	DataFile	DataEn	AppExpr	TeamSize	Method
5	DataEn	DataOut	AppExpr	TeamSize	SAType
6	DataEn	DataOut	ToolExpr	AppExpr	TeamSize
7	DataFile	DataEn	Tools	AppExpr	TeamSize
..	..	..	..	..	..

2.6.4 Application of RSA for Feature Ranking and Selection

The *Reducts* obtained in Section 2.6.3 are used as input to this technique. After getting the reducts, the number of occurrence of each attribute is calculated and denoted as n_i. To calculate the weight of an attribute Equation 2.1 is used. For

example, weight of attribute IntComplx, for USP05-FT data, is calculated as:

$$\begin{aligned} w(IntComplx) &= \frac{\#occurrence\ of\ IntComplx}{\sum_{j=1}^{m} N_j} \\ &= \frac{5}{75} \\ &= 0.0667 \end{aligned}$$

Table 2.4 gives the number of occurrence and the weight of each attribute for both the datasets. By looking at the Table it is clear that there are some attribute which are not part of any of the reduct. So weight for those attribute is zero.

Table 2.4: Weights from reduct using RSA for both the Datasets

	Rank(Attribute)			
	USP05-FT Data		**USP05-RQ Data**	
Attribute Name	**#Occurrence in reducts**	**Weight**	**#Occurrence in reducts**	**Weight**
IntComplx	5	0.0667	10	0.1639
DataFile	5	0.0667	10	0.1639
DataEn	15	0.2000	5	0.0819
DataOut	5	0.0667	10	0.1639
UFP	0	0.0	10	0.1639
Lang	3	0.0400	2	0.0327
Tools	3	0.0400	2	0.0327
ToolExpr	3	0.0400	2	0.0327
AppExpr	15	0.2000	2	0.0327
TeamSize	15	0.2000	5	0.0819
DBMS	0	0.0	1	0.0164
Method	3	0.0400	0	0.0
AppType	3	0.0400	2	0.0327

Once the weight has been calculated, the attributes are sorted in decreasing order according to weight. As higher the weight, the attribute is more relevant and important to calculate the target attribute. The attributes are selected based on some threshold value. To find out the optimal threshold value, the machine learning model is implemented with different values and the one with minimum error gives the optimal threshold value. For example, subset of selected attributes with threshold 0.05, for USP05-FT data is:

{ IntComplx, DataFile, DataEn, DataOut, AppExpr, TeamSize }

2.6.5 Application of Information Gain for Feature Ranking and Selection

Information Gain takes discrete data, result of Section 2.6.2, as input. For each attribute A_i, Info gain is calculated using Equation 2.4. Info gain of an attribute A_i shows the reduction in information required to classify the target attribute by knowing the value of A_i.

Table 2.5 gives the value of info gain for each attribute for both the datasets.

Table 2.5: Info Gain of each attribute for both the Datasets

	Info Gain (Attribute)	
Attribute Name	**USP05-FT Data**	**USP05-RQ Data**
IntComplx	0.6334	0.5235
DataFile	0.5533	0.2213
DataEn	0.3622	0.1820
DataOut	0.0785	0.2969
UFP	0.4698	0.1521
Lang	0.4558	0.5282
Tools	0.4885	0.5183
ToolExpr	0.6324	0.4871
AppExpr	0.2672	0.3777
TeamSize	0.4758	0.2150
DBMS	0.5280	0.1914
Method	0.3875	0.3941
AppType	0.0149	0.01983

Once the info gain has been calculated, the attributes are sorted in decreasing order according to info gain. As higher the info gain, more reduction in required information to calculate the target attribute. In other words it can be said that less information is needed to classify the target attribute. The attributes are selected based on some threshold value. To find out the optimal threshold value, the machine learning model is implemented with different threshold values and the one with minimum error gives the optimal threshold value. For example, subset of selected attributes with threshold 0.5, for USP05-FT data is:

{ IntComplx, DataFile, ToolExpr, DBMS}

2.7 Summary

In this chapter, various feature selection techniques are discussed. The original data is preprocessed and discretization is performed for continuous attributes. Different feature selection technique are implemented on discretized data and most relevant features are selected. The reduced feature set works as input for the next chapter, which is the application of machine learning techniques for software effort estimation.

Chapter 3

Software Effort Estimation using Machine Learning Techniques

3.1 Introduction

Software development effort estimation is one of the most difficult task in software engineering. Most of the projects are failed due to inaccurate estimated effort. So the success of any software project depends on an early and accurate effort estimation. In literature many methods have been proposed by researchers to help the managers of software industry in effort estimation practice. The main focus of this chapter is to estimate the effort of several software projects using machine learning techniques.

In this chapter various machine learning techniques are employed for effort estimation. These techniques can be grouped into two type: Artificial neural network (ANN) and Classification models. In ANN technique, four different networks are considered, as the working of these networks is different. Three different classification models are used in this work, based on the past applications in literature. Furthermore, a comparative analysis for software effort estimation using various machine learning methods has been provided.

3.2 Artificial Neural Network Techniques

ANN is a computational model inspired by the human brain and is a commonly used ML technique in the field of pattern recognition and classification. These

are usually represented as a system of interconnected 'neurons', that can compute values from its inputs by passing information through the network. This section describes different ANN models which are used to estimate software development effort. Other research using ANN techniques for software effort estimation are explained in [27].

3.2.1 Feed Forward Neural Network (FFNN)

In this study, a Feed Forward Neural Network(FFNN) trained with back propagation learning algorithm is used to estimate the effort. To develop FFNN, artificial neurons, also called nodes, are interconnected in the form of layers. The foremost layer is known as the input layer, a set of neurons which take inputs from outside world, the final layer is known as the output layer, sends some output to outside environment, and the layers between those two are known as hidden layers. The association between the i^{th} and j^{th} node is defined by the weight coefficient w_{ij}. All the neurons of one layer produce some output, which acts as input to the next layer. This 'next layer' can be either the hidden layer or the output layer [28]. Figure 3.1 provides a graphical representation of a Feed Forward Neural Network.

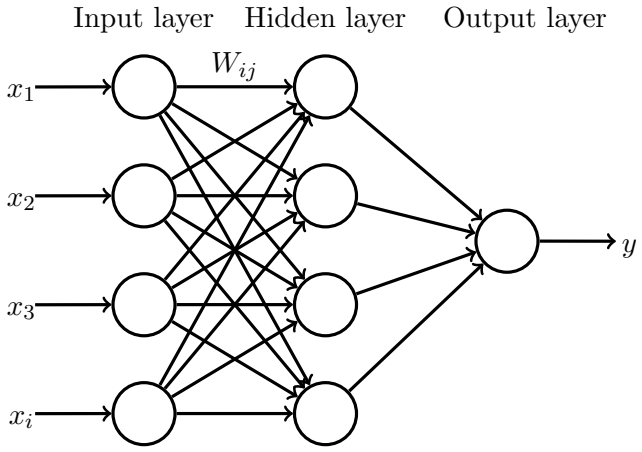

Figure 3.1: Architecture of Feed Forward Neural Network

3.2.2 Radial Basis Function Neural Network (RBFNN)

The radial basis function is a classification and functional approximation neural network. RBFN uses the most common activation functions such as Sigmoidal and Gaussian kernel function. The architecture for the RBFN is shown in Figure 3.2. The architecture consists of hidden units, best known as radial centers, expressed by the vectors $c_1, c_2, ..., c_h$. Hidden neurons offer a set of centers that comprise an absolute basis for the input patterns. The conversion from input layer to hidden unit space is nonlinear whereas conversion from hidden unit to output unit is linear. A significant non-zero outcome will be produced by the hidden layer of radial basis function network, when the input pattern falls inside a little confined area of the data space.

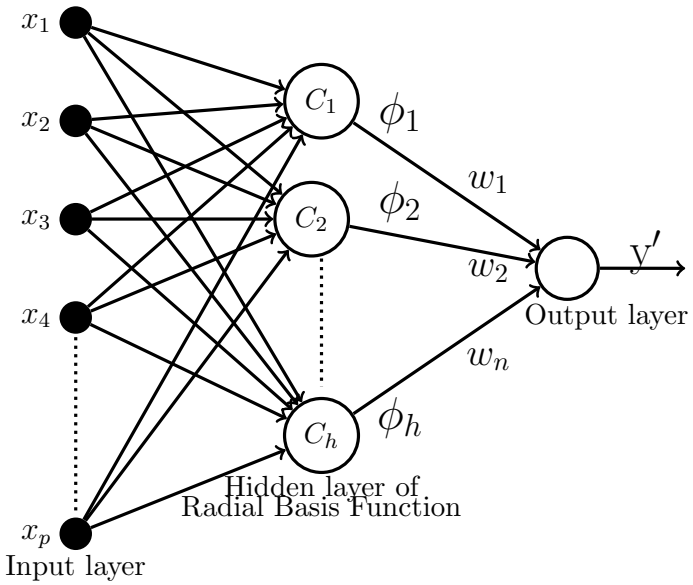

Figure 3.2: Architecture of RBFN network

3.2.3 Functional Link Artificial Neural Network (FLANN)

The FLAN network, for estimating the effort to develop a software project, is a one layer feed forward neural network. It comprises of one input layer and one output layer. The FLAN model computes the network output by extending the initial inputs and after that serving it to the final output layer, as given by Yoh Han Pao in 1989 [29].

In the perspective of learning, the FLAN system will be much quicker than other neural system. The essential reason behind the efficiency of FLAN is that the learning process in FLAN network has two phases and both the phases can be made effective by appropriate learning process. The architecture of FLANN is presented in Figure 3.3. In this study one out of three distinctive functional extension of the input object in the FLAN network has been performed. These are Chebyshev, Legendre and Power Series expansion, named as C-FLANN, L-FLANN and P-FLANN respectively. C-FLANN has been used in this work.

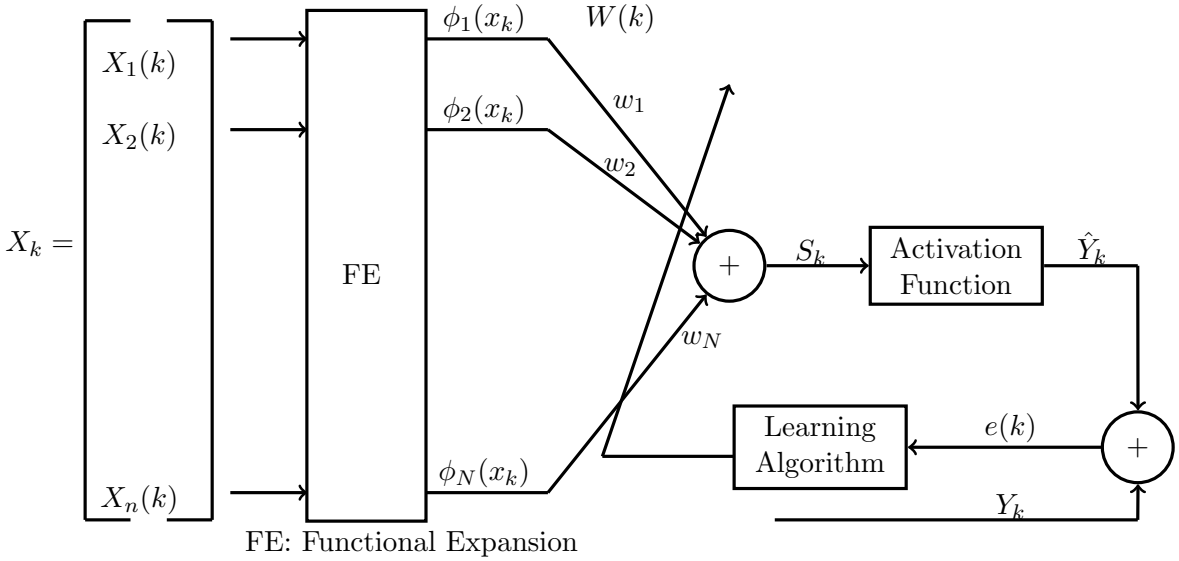

Figure 3.3: A typical Functional Link Artificial Neural Network

- The **Chebyshev** expansions are expressed as:

  $C_0(x) = 1, \qquad C_1(x) = x, \qquad C_2(x) = 2x^2 - 1,$
  $C_3(x) = 4x^3 - 3x, \qquad C_4(x) = 8x^3 - 8x^2 + 1$

  More higher order Chebyshev polynomials might be produced by utilizing the recursive equation given as:

$$C_n(x) = 2xC_{n-1}(x) - C_{n-2}(x), n \geq 2, (-1 \leq x \leq 1) \tag{3.1}$$

3.2.4 Levenberg Marquadt Neural Network (LMNN)

The Levenberg-Marquardt (LM) algorithm, was introduced by Kenneth Levenberg and Donald Marquardt [30]. The fundamental thought of the LM procedure is

that a composed training process has been performed by this. The LM algorithm follows two operations. It applies the steepest descent operation, until the local curvature is fitting to make a quadratic approximation, then it applies the Gauss-Newton operation, which can speed up the convergence importantly. Figure 3.4 presents the flow chart of LM neural network.

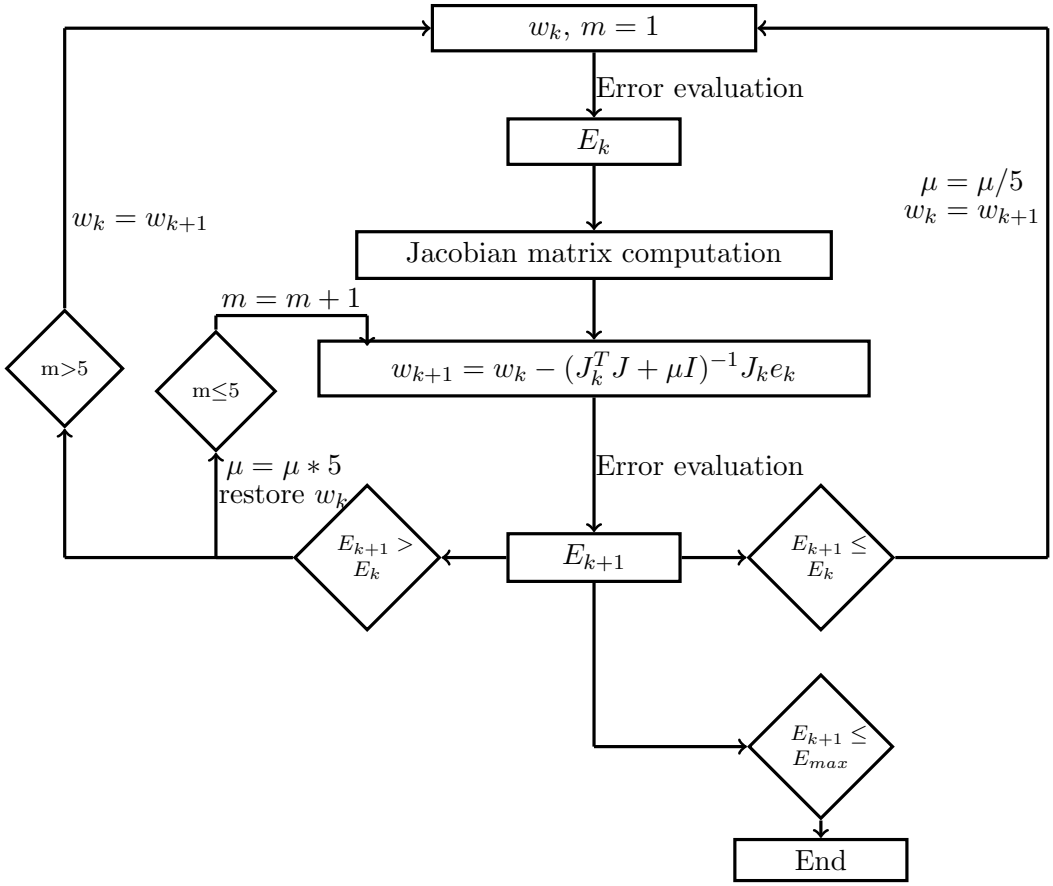

Figure 3.4: Overview of Estimation Process using LM Neural Network

Steps to be followed for LM neural network are given here:

1. By considering the initial weights (which are generated randomly), compute the total error (RMSE).

2. The weights are updated using the Equation 3.2 as:

$$w_{k+1} = w_k - (J_k^T J + \mu I)^{-1} J_k e_k \tag{3.2}$$

3. Taking the new weights, again evaluate RMSE.

4. If the RMSE is goes up as an aftereffect of the update, then withdraw the step and increase the combination coefficient μ by an element of 5. At that point go to step 2 and try an update again.

5. If the RMSE is diminished as an aftereffect of the update, then acknowledge the step and decrease the combination coefficient μ an element of 5, same as step 4.

6. Move to step 2 with the new network weights until the RMSE is less than the requisite value.

3.3 Classification Techniques

3.3.1 Naive Bayes Classifier (NBC)

Naive Bayes classifier also known as Bayesian classification method and is founded on Bayes′ theorem. It assumes that all the features are independent in nature and will not affect the estimation process [31]. Figure 3.5 shows a Naive Bayes classifier, in which all arcs are directed from class attribute to all other attributes.

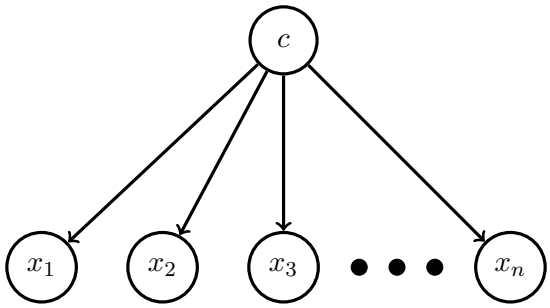

Figure 3.5: A simple Naive Bayes classifier

Consider D be a set of input objects with their marked class labels. Assume X be a single object in set. In Bayesian theory, X is conceived as *evidence*. Each object is expressed by an n-dimensional feature vector, $X = (x_1, x_2, ..., x_n)$, rendering n observations made on the object from n features, respectively, $A_1, A_2, ..., A_n$. Assume that m classes are there in input data, $C_1, C_2, ..., C_m$. The goal is to find the likelihood that object X be a member of class C_i, given the feature description of X. Working of Naive Bayes Classifier is described here:

The Naive Bayes classifier allots the given object x to class $c^* = argmax_c P(c|x)$ by using *Bayes'* rule presented below:

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)} \tag{3.3}$$

as $P(x)$, *prior probability* of x, is invariant for all classes so it plays no role is selecting c^*, and $P(x|c)$ is referred to as *conditional probability* is given as:

$$P(x|c) = \prod_{k=1}^{n} P(x_k|c) \tag{3.4}$$

A little issue with the model is that if a given class and characteristic esteem never happens together in the preparation set, then the occurrence based likelihood assessment will be zero. The consequences of this condition will be visible when all information in the other likelihood are multiplied. Hence, *Laplacian correction* or Laplace estimator is used.

3.3.2 Classification And Regression Tree (CART)

CART, first brought in by Breiman et al. [32], is a decision tree which can be used for both purpose, classification and regression. In this work it is used as a classification model. The classification tree is a binary decision tree, build up from class marked training objects. The classification tree has two kind of nodes: 1. Internal (Non-leaf) node represents the condition to be tested for an attribute. 2. Terminal (leaf) node gives a class label. Algorithm to construct the classification tree works in top-down manner and consist of three step.

i. Selection of the most significant attribute and its value, to partition the data at each level.

ii. Selecting the threshold value to stop partitioning.

iii. Selection of the optimal tree with least testing error.

CART uses *Gini Index*, given in Equation 3.5, as splitting criteria to find the best splitting attribute.

$$Gini(D) = 1 - \sum_{i=1}^{m} p_i^2 \tag{3.5}$$

At each level of tree, Gini index is calculated for every attribute and the one with the minimal Gini index is chosen for the splitting attribute. The final tree is used as a classifier to forecast the class label for a new project.

3.3.3 Support Vector Classification (SVC)

Support vector machine (SVM) is a concept in statistical learning theory and a nonlinear machine learning technique. SVM is applicable for both classification and regression problems. In this study it is used as a classification model based on supervised learning.

For a two class problem, SVC tries to find a hyperplane which separates the objects of both classes. The hyperplane is defined by the 'support vectors', the most important training object in data. To handle inseparable and non-linear data, SVC uses kernel functions to interpret the non-linear data into high dimensional space which is then linearly separable.

In this study SVC is considered for software effort estimation with multi-class data. There are two ways to perform multi-class classification using SVC: One-vs-All and One-vs-One. The One-vs-All approach build N binary classifiers for N class problem. Each of the SVC separates a single class from all remaining classes. For the i^{th} classifier, the new pattern is classified as either a member of class i or not a member of class i. In the one-vs-one technique, also known as pair-wise approach, $\frac{n(n-1)}{2}$ binary classifiers are built, where every SVC is skilled on objects from two classes [33].

3.4 Proposed Approach

This section emphasizes on effort estimation by above mentioned techniques for software development. The main stages for setting up an estimation by proposed techniques are presented in Figure 3.6:

Steps in Software Effort Estimation

1. **Data Selection**: data about related projects undertaken in past are collected. In this work modified data after performing feature selection tech-

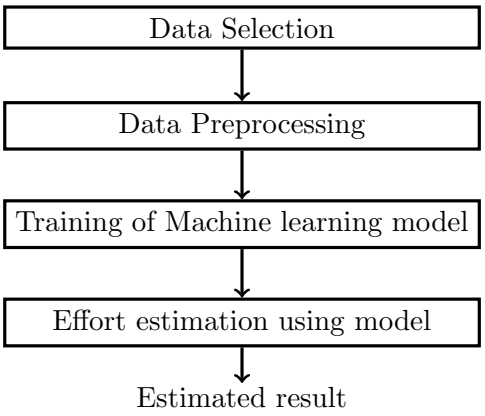

Figure 3.6: Framework of Effort estimation model

niques are used as input to machine learning models.

2. **Data Preprocessing**: data is preprocessed if necessary.

3. **Application of Machine learning models**: Estimation model is being tuned with reduced data. Tunning can have strong impact on the quality of estimation.

4. **Effort Estimation**: Effort is estimated for a new project using the estimation model.

3.5 Implementation and Results

In this section various machine learning techniques are implemented with the result of different feature selection techniques. Three feature selection technique are considered in this work and each technique is paired with every machine learning model. A comparative analysis is performed using various performance measure described in Section 1.4.

3.5.1 Application of ANN Techniques

ANN models are trained with reduced dataset for estimating the effort of a new project. The following section shows the application of each neural network one by one.

3.5.1.1 Data Preprocessing: Normalization

As in ANN, the range of input values is in between $(-1, 1)$, the data need to be normalized within this range. To normalize a set of data, the original data range is mapped into another scale. In the work presented here, the input data is being normalized in the range of $(0, 1)$. Many normalization methods are available in literature [34]. Among them *min-max normalization* method is used in this work as follows:

For a data vector $X(x_1, x_2, x_3....., x_n)$

i. the highest value of actual data set, $orgMax$ is found out

ii. the minimal value of actual data set, $orgMin$ is found out

iii. the maximum and minimum value of the normalized scale are termed as $newMax$ and $newMin$ respectively.

iv. For any value x_i the normalized y_i is computed as

$$y_i = \frac{x_i - orgMin}{orgMax - orgMin}(newMax - newMin) + newMin \tag{3.6}$$

The normalized data is given as input to ANN to train the model.

3.5.1.2 Application of Feed Forward Neural Network

FFNN model is created with one hidden layer, five (six for USP05-RQ) nodes in the input layer and one output node. To train the network, *backpropagation* algorithm has been used [35] and *5-fold Cross Validation* learning is performed in this study.

The normalized data are fed into the input layer. The input data is multiplied with the weights and the result of multiplication from all the neurons connected to the hidden layer neuron j are summed. The summed result is fed into the activation function to get the input to the neuron j. The *Sigmoid* activation function, given below, is used where $f(x_j)$ is the net input for neuron j.

$$f(x_j) = \frac{1}{1 + exp^{-x_j}} \tag{3.7}$$

Output layer neurons work in the same way to generate the outcome of the output layer. The net value from the activation function of the final layer is the expected deviation for the given training objects. The network outcome is contrasted with the actual output to find out the error for i^{th} training object using following equation:

$$error_i = |actualOutput_i - computedOutput_i| \tag{3.8}$$

The weights of networks are updated based on this error and the complete process is repeated for every data point. When all data points in the training set are considered, one epoch is completed.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} error_i^2}{N}} \tag{3.9}$$

For a training set having N number of data point, in every epoch, root mean square error(RMSE) is calculated using equation 3.9. This whole process is repeated until either RMSE reaches a threshold value (0.001 in this work) or epochs crosses the limit of maximum number of epochs (2000 in this work). Learning rate and momentum coefficient for FFNN are 0.8 and 0.1 respectively.

Table 3.1 presents the outcome of FFNN for USP05-FT for three feature selection techniques. The maximum number of epochs is 2000. Results of FFNN for USP05-RQ data is given in Table 3.2.

Table 3.1: Result of FFNN for USP05-FT data

	USP05-FT data – FFNN				
	No. of hidden neurons	**Training RMSE**	**Test RMSE**	**Test MMRE**	**Test MAE**
RSA Reduct	6	0.1918	0.1569	0.1471	0.1039
RSA Rank	11	0.1687	0.1805	0.1344	0.1217
Info Gain	8	0.1740	0.2094	0.1518	0.1239

Figure 3.7 and Figure 3.8 show a graphical representation of actual and estimated values of effort for both the dataset respectively. The USP05-FT data set used for FFNN is Reduct No.3. of Table 2.3 obtained on applying rough set analysis.

Table 3.2: Result of FFNN for USP05-RQ data

	USP05-RQ data – FFNN				
	No. of hidden neurons	**Training RMSE**	**Test RMSE**	**Test MMRE**	**Test MAE**
RSA Reduct	7	0.2377	0.2632	0.1526	0.1943
RSA Rank	10	0.2145	0.2430	0.1759	0.1455
Info Gain	7	0.1632	0.1896	0.1786	0.1384

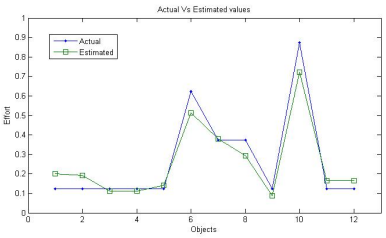

Figure 3.7: Actual vs Estimated Effort using FFNN for USP05-FT data

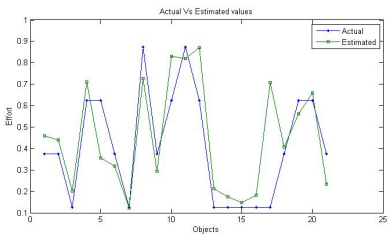

Figure 3.8: Actual vs Estimated Effort using FFNN for USP05-RQ data

3.5.1.3 Application of Radial Basis Function Network

A sufficient number of centers are chosen from the set of training objects. The yield from the hidden layer unit is computed using Equation:

$$y_i\left(x_i\right) = \frac{exp[-\sum_{j=l}^{r}(x_{ji}-c_{ji})^2]}{\sigma_i^2}$$

where c_{ji} is the center of the RBF unit for input objects; σ_i the width of the i^{th} RBF unit. The output of the neural network is calculated using below Equation:

$$y_{net} = \sum_{i=1}^{k} w_i y_i(x_i)$$

where k shows the number of hidden layer nodes. Compute deviation for n^{th} training set using following Equation:

$$error_n = |actualOutput_n - computedOutput_n|$$

The weights of networks are updated based on this error. Training stops when RMSE reaches a threshold value.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} error_i^2}{N}}$$

The threshold value of weight for Ranked_RSA data is 0.05 and threshold gain value is 0.4 for RBFN. Table 3.3 and Table 3.4 show the results of RBFN for USP05-FT and USP05-RQ datasets respectively, where the value of the maximum number of epochs is 2000.

Table 3.3: Result of RBFN for USP05-FT data

	USP05-FT data – RBFN				
	No. of cluster	**Training RMSE**	**Test RMSE**	**Test MMRE**	**Test MAE**
RSA Reduct	4	0.2576	0.2826	0.2050	0.2311
RSA Rank	4	0.2937	0.2800	0.1765	0.2171
Info Gain	4	0.2569	0.2788	0.2131	0.2567

Table 3.4: Result of RBFN for USP05-RQ data

	USP05-RQ data – RBFN				
	No. of cluster	**Training RMSE**	**Test RMSE**	**Test MMRE**	**Test MAE**
RSA Reduct	4	0.3015	0.3765	0.2273	0.2878
RSA Rank	4	0.3248	0.3914	0.2941	0.3429
Info Gain	4	0.3566	0.4251	0.3162	0.3940

Figure 3.9 and Figure 3.10 present a graphical view of the actual and the estimated effort for both datasets respectively.

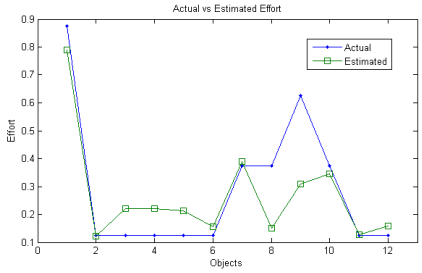

Figure 3.9: Actual vs Estimated Effort using RBFN for USP05-FT data

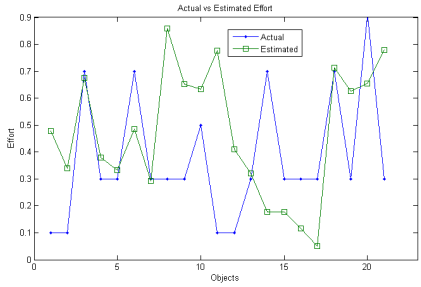

Figure 3.10: Actual vs Estimated Effort using RBFN for USP05-RQ data

3.5.1.4 Application of Functional Link Artificial Neural Network

Expansion of each input using *Chebyshev* polynomials is exercised as follows:

$C_0(x) = 1, \quad C_1(x) = x, \quad C_2(x) = 2x^2 - 1, \quad C_3(x) = 4x^3 - 3x$

$C_4(x) = 8x^3 - 8x^2 + 1$

The extended input is multiplied by weights and after that summed to find out the estimated output $\hat{y}(k)$ and error $e(k)$.

$$\hat{y}(k) = \sum_{j=1}^{J} Tf_j(k) * w_j(k)$$

$$e(k) = y(k) - \hat{y}(k)$$

Calculate change in weight for each input for k^{th} iteration and average change in weight for P objects as:

$$\Delta w_j(k) = \mu * Tf_j(k) * e(k)$$

$$\overline{\Delta w_j(k)} = \frac{1}{P} \sum_{i=1}^{P} \Delta w_j^i(k)$$

The weights are updated using below Equation:

$$w_j(k+1) = w_j(k) + \Delta w_j(k)$$

where, $w_j(k)$ is the j^{th} weight at the k^{th} iteration and μ is the convergence coefficient, the value of μ lies between 0 to 1. J is defined as $J = m * d$ and $1 < j < J$.

Table 3.5 presents the results of C-type FLANN for USP05-FT data where the optimal number of expanded node for each input is given for each feature selection technique. Results of FLANN for USP05-RQ data are given in Table 3.6.

Table 3.5: Result of FLANN for USP05-FT data

	USP05-FT data – FLANN				
	No. of expanded input	**Training RMSE**	**Test RMSE**	**Test MMRE**	**Test MAE**
RSA Reduct	4	0.2159	0.2329	0.0930	0.1473
RSA Rank	3	0.1681	0.1865	0.1141	0.1205
Info Gain	5	0.1821	0.2201	0.1263	0.1376

Table 3.6: Result of FLANN for USP05-RQ data

	USP05-RQ data – FLANN				
	No. of expanded input	**Training RMSE**	**Test RMSE**	**Test MMRE**	**Test MAE**
RSA Reduct	5	0.3562	0.4032	0.1586	0.1970
RSA Rank	2	0.2985	0.3313	0.1689	0.1831
Info Gain	4	0.3521	0.4115	0.1660	0.1852

Figure 3.11 and Figure 3.12 show a graphical representation of the effect of the number of input expansion on MMRE for both the datasets respectively for RSA-Reduct feature selection technique.

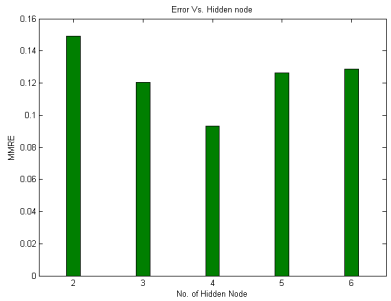

Figure 3.11: MMRE vs. Expanded node for FLANN USP05-FT

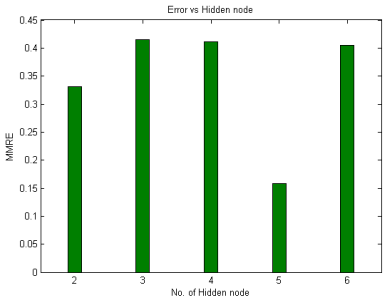

Figure 3.12: MMRE vs. Expanded node for FLANN USP05-RQ

3.5.1.5 Application of Levenberg Marquadt Neural Network

The steps to model a Levenberg Marquadt neural network are given in Section 3.2.4. For every feature selection technique, a LMN network is modeled by following those steps. The value of μ, smoothing parameter, is varied to find an optimal value based on the error with each model.

Table 3.7 presents the results of LMNN for USP05-FT data where the number of classes is four and value of smoothing parameter is also given. Results of LMNN for USP05-RQ data are given in Table 3.8.

Table 3.7: Result of LMNN for USP05-FT data

	USP05-FT data – LMNN				
	μ	**Training RMSE**	**Test RMSE**	**Test MMRE**	**Test MAE**
RSA Reduct	0.5	0.2237	0.2575	0.1779	0.1827
RSA Rank	0.3	0.2532	0.2747	0.1956	0.2059
Info Gain	0.2	0.2739	0.3105	0.2324	0.2363

Table 3.8: Result of LMNN for USP05-RQ data

	USP05-RQ data – LMNN				
	μ	**Training RMSE**	**Test RMSE**	**Test MMRE**	**Test MAE**
RSA Reduct	0.3	0.2136	0.2627	0.1550	0.2189
RSA Rank	0.1	0.1736	0.2143	0.1641	0.1745
Info Gain	0.2	0.3014	0.2818	0.1702	0.2336

Figure 3.13 and Figure 3.14 show a graphical representation of the Effect of μ on MMRE for both the datasets respectively for RSA-Reduct feature selection technique. Value of μ for which MMRE is least is selected for network.

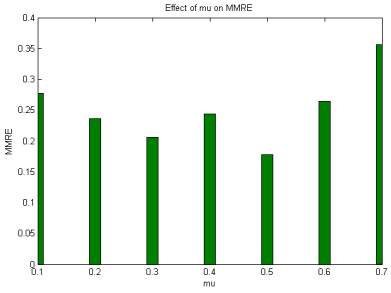

Figure 3.13: Effect of μ on MMRE for USP05-FT data

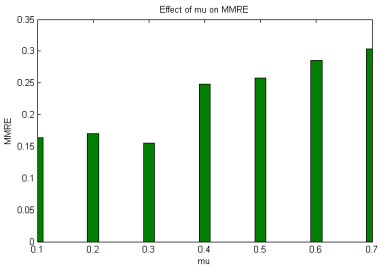

Figure 3.14: Effect of μ on MMRE for USP05-RQ data

3.5.1.6 Comparison of ANN Techniques

Each artificial neural network is trained and tested with three feature selection techniques. The results for each ANN are given for both the datasets. A comparative analysis has been performed in order to find an optimal pair of ANN and feature selection technique.

Figure 3.15 shows a bar graph representation of the comparative study of all ANN's with each feature selection technique for USP05-FT data. It is visible from figure that results obtained from FLAN network are better as compared with other networks. One more analysis that can be drawn from figure is that all ANN's work better with RSA than info gain technique used in the study.

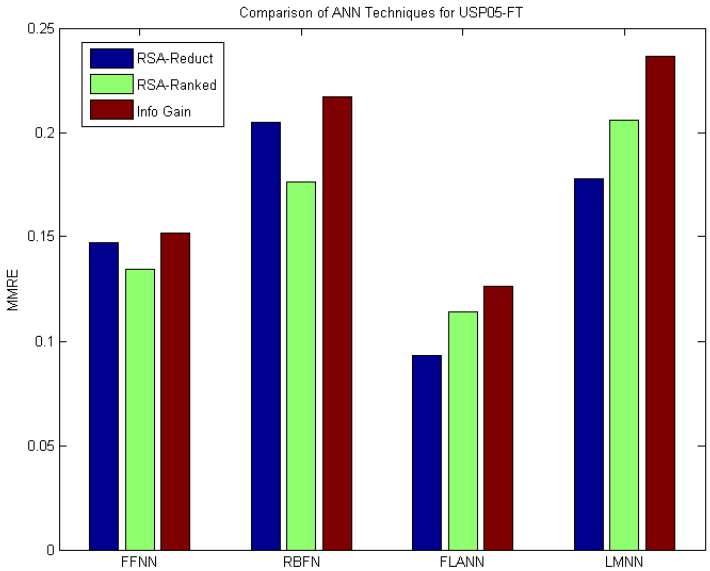

Figure 3.15: Comparison of MMRE for various ANN techniques for USP05-FT

Figure 3.16 shows a comparative analysis of training and testing error obtained using various artificial neural networks for RSA-Reduct feature selection technique. The results are for USP05-FT data and the performance parameter is RMSE.

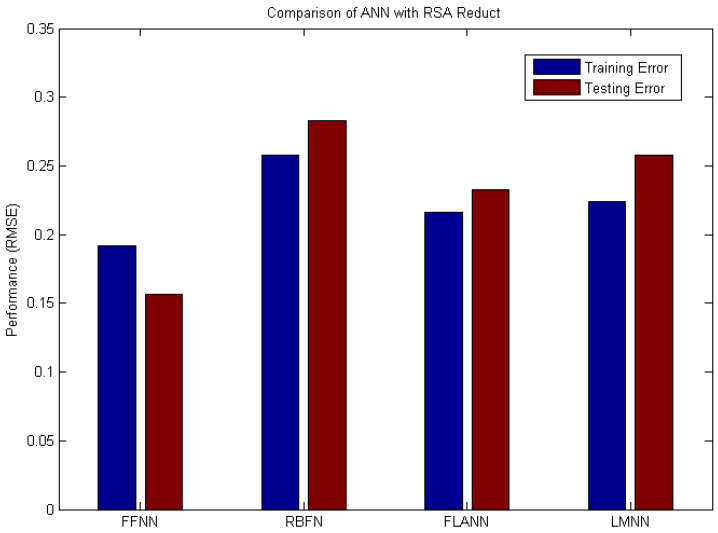

Figure 3.16: Comparison of Error values obtained from training and test set using RSA Reduct and various ANN techniques for USP05-FT

Figure 3.17 shows a comparative analysis of training and testing error obtained using various artificial neural networks for RSA-Rank feature selection technique. The results are for USP05-FT data and the performance parameter is RMSE.

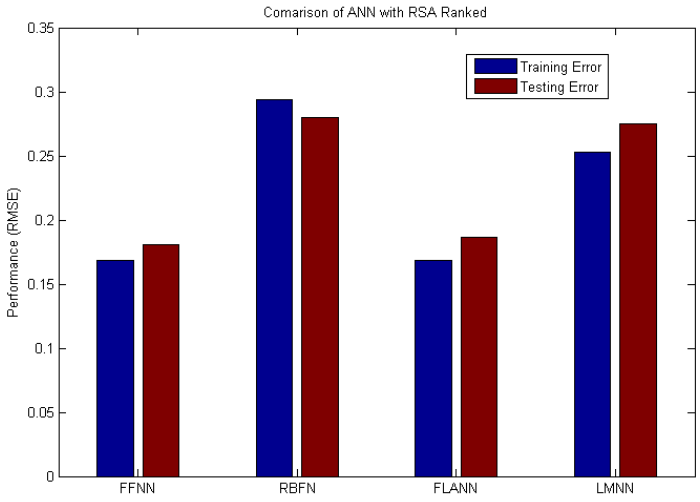

Figure 3.17: Comparison of Error values obtained from training and test set using RSA Ranked and various ANN techniques for USP05-FT

Figure 3.18 shows a comparative analysis of training and testing error obtained

using various artificial neural networks for Info Gain feature selection technique. The results are for USP05-FT data and the performance parameter is RMSE.

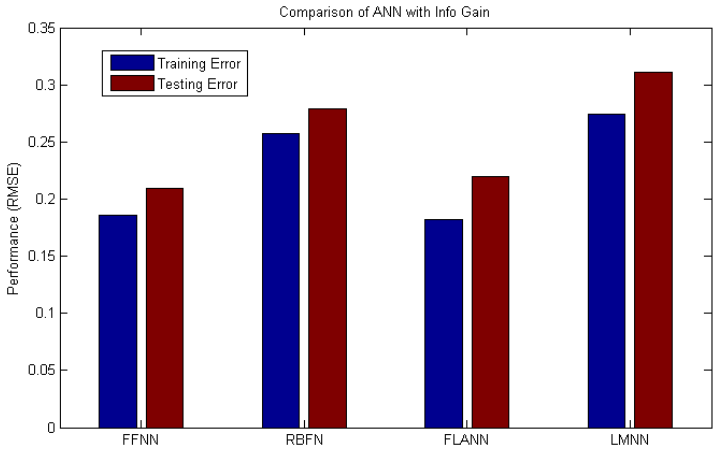

Figure 3.18: Comparison of Error values obtained from training and test set using Info Gain and various ANN techniques for USP05-FT

Figure 3.19 shows a bar graph representation of the comparative study of all ANN's with each feature selection technique for USP05-RQ data. It is visible from figure that results obtained from FFNN network are better as compared with other networks. One more analysis that can be drawn from figure is that all ANN's work better with RSA-Reduct than other feature selection techniques used in the study.

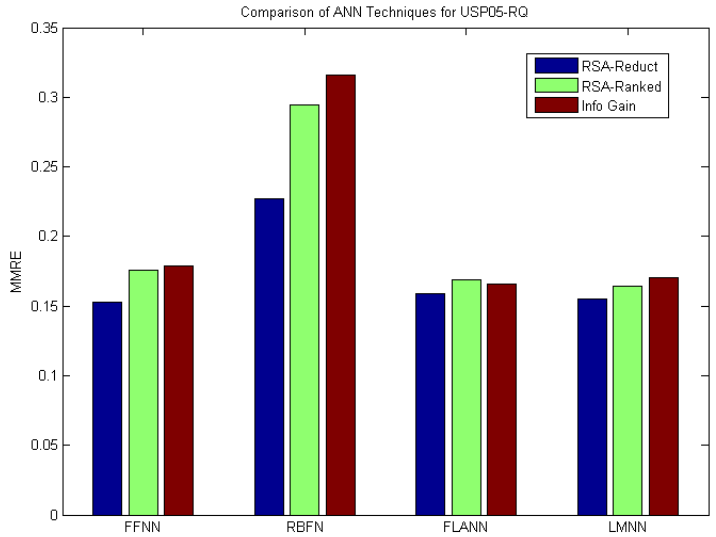

Figure 3.19: Comparison of MMRE for various ANN techniques for USP05-RQ

3.5.2 Application of Classification Techniques

3.5.2.1 Application of Naive Bayes Classifier

The Naive Bayes classifier used in this study is based on the mathematical manipulation of the probabilities. Once the *prior, conditional* and *posteriori* probabilities for each class have been calculated, the *posteriori* probability values are compared to each other. The class with the most likelihood is allocated to the object being classified. Throughout the training stage, Leave-One-Out-Cross-Validation (or Jack-knife cross-validation) based learning is performed to focus on the estimation accuracy distribution. The framework of Naive Bayes classifier is demonstrated in Figure 3.20.

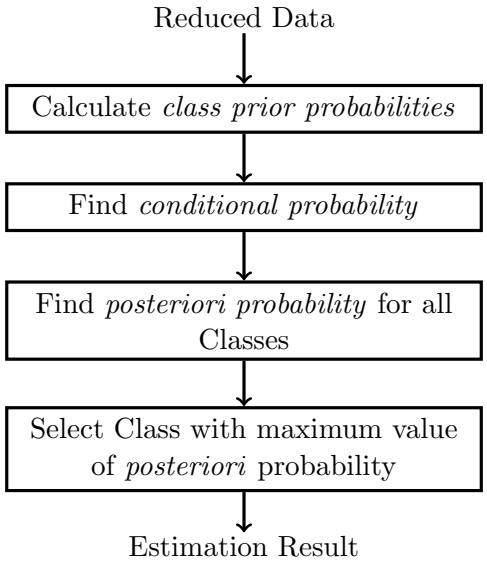

Figure 3.20: Framework of Naive Bayes Classifier for effort estimation

Table 3.9 and Table 3.10 show the results of Naive Bayes classifier for USP05-FT and USP05-RQ dataset respectively. *Jack-Knife* cross fold validation is used to find the accuracy of each classification technique.

Table 3.9: Results of Naive Bayes classifier for USP05-FT

	Precision	**Recall**	**Accuracy**
RSA Reduct	0.8495	0.8377	0.8947
RSA Rank	0.7438	0.7465	0.8245
Info Gain	0.5476	0.5292	0.8070

Table 3.10: Results of Naive Bayes classifier for USP05-RQ

	Precision	Recall	Accuracy
RSA Reduct	0.8074	0.8204	0.8039
RSA Rank	0.7402	0.7504	0.7352
Info Gain	0.6442	0.6798	0.6470

Figure 3.21 and Figure 3.22 show the graphical representation of actual and estimated values of effort for both dataset respectively. The input of the NBC is the reduct obtained after application of rough set analysis.

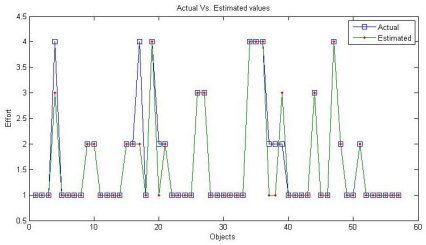

Figure 3.21: Actual vs Estimated Effort using NBC for USP05-FT data

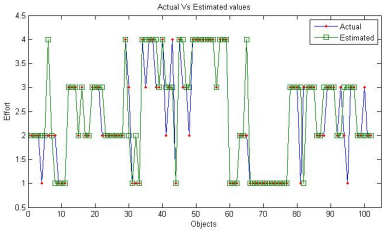

Figure 3.22: Actual vs Estimated Effort using NBC for USP05-RQ data

3.5.2.2 Application of Classification And Regression Tree

CART is been used in this study as a classifier. To find the splitting attribute at each level, Gini index criteria is used. Gini index of each attribute is calculated using Equation 3.5 and the one with the least value of Gini index is selected as splitting attribute. CART has been implemented with the reduced dataset one by one.

The result of the application of the classification tree with the output of the three feature selection techniques is given in Table 3.11 and 3.12 for both the datasets respectively.

Table 3.11: Results of CART for USP05-FT

	Precision	Recall	Accuracy
RSA Reduct	0.8375	0.7543	0.8777
RSA Rank	0.5723	0.5807	0.8245
Info Gain	0.6473	0.6234	0.8596

Figure 3.23 and Figure 3.24 show the graphical view of actual versus estimated

Table 3.12: Results of CART for USP05-RQ

	Precision	Recall	Accuracy
RSA Reduct	0.8126	0.7817	0.7941
RSA Rank	0.7535	0.7343	0.7254
Info Gain	0.7491	0.7261	0.7450

effort using CART, where the input data is the reduct obtained after application of rough set analysis, for both datasets.

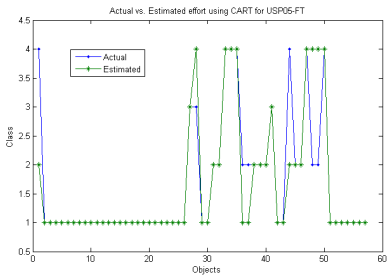

Figure 3.23: Actual vs Estimated Effort using CART for USP05-FT data

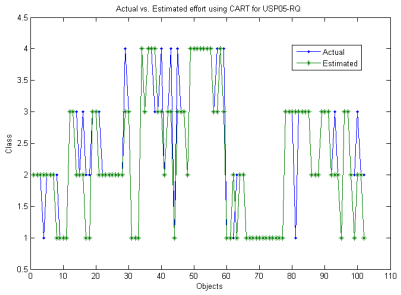

Figure 3.24: Actual vs Estimated Effort using CART for USP05-RQ data

3.5.2.3 Application of Support Vector Classifier

The problem of software effort estimation in this study is of type multi-class problem. There are two ways to solve this kind of problem. The current work handle the problem by using *one-vs-all* approach of multi-class support vector classifier. Four classifier are modeled, as there are four classes in dataset, one for each class. The input to SVC is the datasets obtained after the application of various feature selection techniques.

The results of the application of the support vector classification with the output of the three feature selection techniques are given in Table 3.13 and 3.14 for both the datasets respectively.

Table 3.13: Results of Support Vector classifier for USP05-FT

	Precision	Recall	Accuracy
RSA Reduct	0.6495	0.6818	0.8535
RSA Rank	0.5974	0.4935	0.7894
Info Gain	0.6037	0.5519	0.8245

Table 3.14: Results of Support Vector classifier for USP05-RQ

	Precision	Recall	Accuracy
RSA Reduct	0.7878	0.7877	0.7843
RSA Rank	0.7326	0.7338	0.7254
Info Gain	0.7483	0.7496	0.7450

3.5.2.4 Comparison of Results of Classification Techniques

Each classification model is trained and tested with the output of three feature selection techniques. *Jack-Knife* cross fold validation method has been used in the work. The results for each classification model are given for both the datasets. A comparative analysis has been performed in order to find an optimal pair of classification and feature selection technique.

Figure 3.25 shows a graphical representation of the comparative study of all classification models with each feature selection technique for USP05-FT data in terms of accuracy. Higher the value of accuracy, the model is more efficient. It is visible from the figure that results obtained from Naive Bayes classifier are better as compared with other models. One more observation that can be made from the figure is that all classification models work better with RSA-Reduct than other feature selection techniques used in the study.

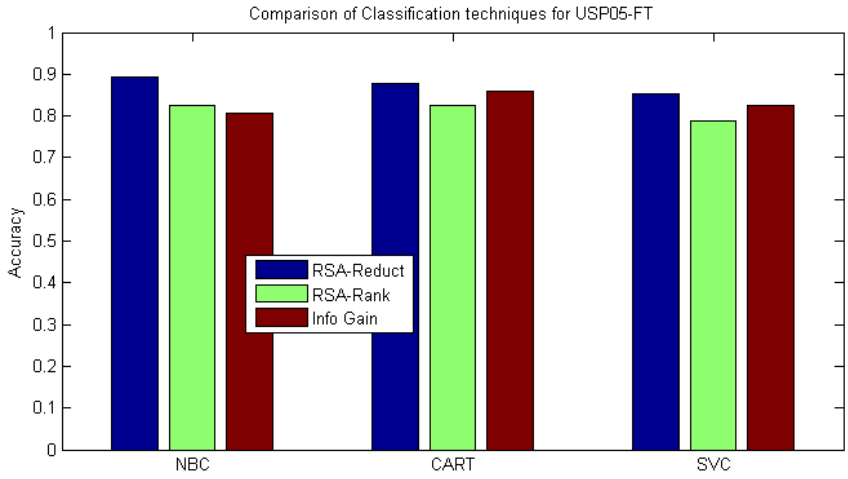

Figure 3.25: Comparison of Accuracy for various Classification techniques for USP05-FT

Figure 3.26 shows a graphical representation of the comparative study of all

classification models with each feature selection technique for USP05-RQ data in terms of accuracy. Again it is evident from the figure that results obtained from Naive Bayes classifier with RSA-Reduct technique are better as compared with other models.

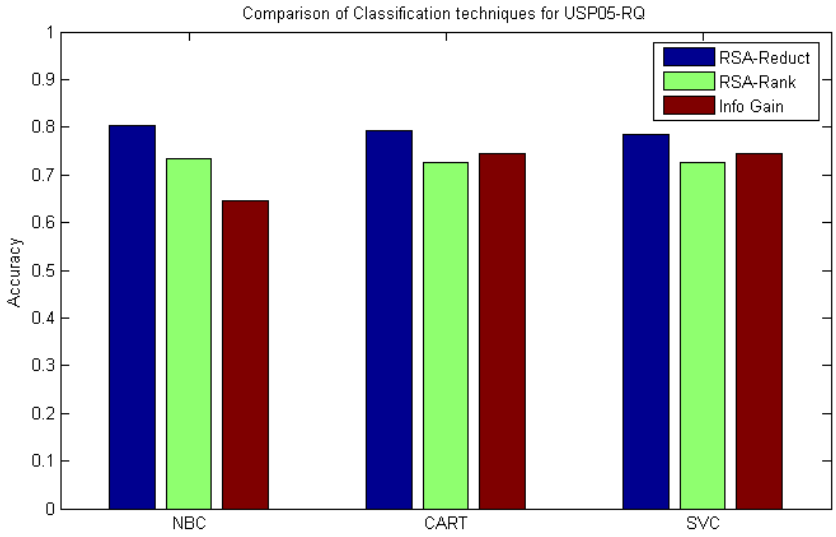

Figure 3.26: Comparison of Accuracy for various Classification techniques for USP05-RQ

3.6 Summary

In literature many techniques have been presented to perform effort estimation. The current work proposed some machine learning techniques for software effort estimation. Various models have been designed with combination of feature selection technique. The efficiency of each model is evaluated using various performance measures. In the last, a comparative analysis is being done in order to find the best combination of feature selection and machine learning technique.

Chapter 4

Conclusion and Future Work

4.1 Conclusion

Accurate estimation of software development effort is one of the most challenging problem in industry. A number of methods have already been proposed in the literature to solve this problem. The approach defined in this study has been carried out in two phases. Two datasets of USP05 from PROMISE software engineering repository have been considered for software effort estimation. In the first phase of approach three feature selection techniques, such as Rough-Reduct, RSA-Rank and Info Gain, are applied to the dataset to find the optimal feature set. The second phase concentrates on the effort estimation for reduced dataset using various machine learning techniques.

The machine learning techniques such as FFNN, RBFN, FLANN, LMNN, NBC, CART and SVC are implemented with the results of three feature selection techniques. In the last, the results are compared in order to find a pair of feature selection and machine learning technique which gives better result than that of other combinations. The above mentioned techniques are implemented using MATLAB.

Figure 4.1 shows a graphical representation of the comparative study of various feature selection methods with ANN machine learning techniques for USP05-FT data. The performance parameter is MMRE for comparison. It is visible from the figure that results obtained from FLAN network are better as compared with other networks. One more observation that can be made from the figure is that

with each feature selection output FLAN gives good result. The second network which performs better is FFNN. Among the feature selection methods, the RSA method performs better than Info gain method.

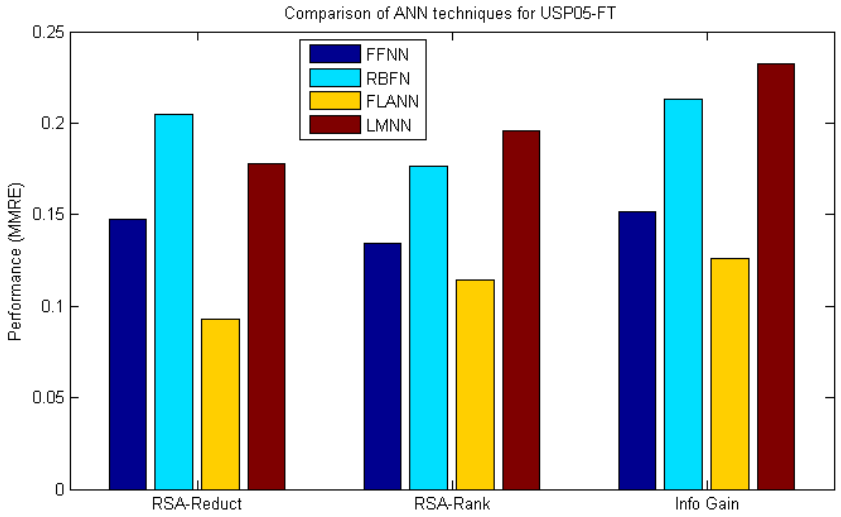

Figure 4.1: Comparison of performance for various feature selection and ANN techniques for USP05-FT

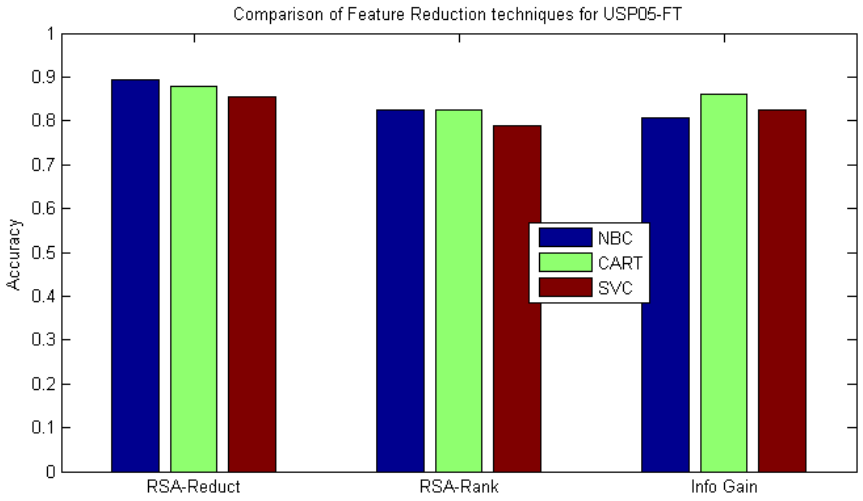

Figure 4.2: Comparison of Accuracy for various feature selection and Classification techniques for USP05-FT

Figure 4.2 shows a graphical representation of the comparative study of various feature selection methods with Classification machine learning techniques for USP05-FT data. Here the performance parameter is accuracy for comparison.

It is evident from the figure that results obtained from RSA-Reduct are better as compared with the rest two feature selection technique. One more observation that can be made from the figure is that the Naive Bayes classifier performs better with each feature selection output.

4.2 Future Work

The study can be extended to simulate the proposed approach using other feature selection techniques such as relief, wrapper subset selection and T-test method. The efficiency of these methods can be found out using many classification techniques. The strength of artificial intelligence techniques like genetic algorithm, particle swarm optimization (PSO) and fuzzy logic can also be used for software effort estimation. The work can be further carried out by finding a combination of feature selection and classification technique which gives optimal results.

Bibliography

[1] B. W. Boehm, "Software engineering economics," *Software Engineering, IEEE Transactions on*, vol. 10, no. 1, pp. 4–21, January, 1984.

[2] A. J. Albrecht and J. E. Gaffney Jr, "Software function, source lines of code, and development effort prediction: a software science validation," *Software Engineering, IEEE Transactions on*, vol. 9, no. 6, pp. 639–648, November, 1983.

[3] K. Srinivasan and D. Fisher, "Machine learning approaches to estimating software development effort," *Software Engineering, IEEE Transactions on*, vol. 21, no. 2, pp. 126–137, February, 1995.

[4] J. Li and G. Ruhe, "Usp05-ft: Software effort estimation at feature and requirement levels." `http://promisedata.org/repository`, December 2005.

[5] Z. Pawlak, "Rough set approach to knowledge-based decision support," *European journal of operational research*, vol. 99, no. 1, pp. 48–57, May, 1997.

[6] B. S. Ahn, S. S. Cho, and C. Y. Kim, "The integrated methodology of rough set theory and artificial neural network for business failure prediction," *Expert Systems with Applications*, vol. 18, no. 2, pp. 65–74, February, 2000.

[7] G. P. Zhang, "Neural networks for classification: a survey," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on.*, vol. 30, no. 4, pp. 451–462, November, 2000.

[8] L. H. Putnam, "A general empirical solution to the macro software sizing and estimating problem," *Software Engineering, IEEE Transactions on*, vol. 4, no. 4, pp. 345–361, July, 1978.

[9] M. Shepperd and C. Schofield, "Estimating software project effort using analogies," *Software Engineering, IEEE Transactions on*, vol. 23, no. 11, pp. 736–743, November, 1997.

[10] B. Boehm, C. Abts, and S. Chulani, "Software development cost estimation approachesa survey," *Annals of Software Engineering*, vol. 10, no. 1-4, pp. 177–205, November, 2000.

[11] R. Bhatnagar, V. Bhattacharjee, and M. K. Ghose, "Software development effort estimation neural network vs. regression modeling approach," *International Journal of Engineering Science and Technology*, vol. 2, no. 7, pp. 2950–2956, 2010.

[12] P. C. Pendharkar, G. H. Subramanian, and J. A. Rodger, "A probabilistic model for predicting software development effort," *Software Engineering, IEEE Transactions on*, vol. 31, no. 7, pp. 615–624, July, 2005.

[13] A. L. Oliveira, "Estimation of software project effort with support vector regression," *Neurocomputing*, vol. 69, no. 13, pp. 1749–1753, February, 2006.

[14] J. Li, G. Ruhe, A. Al-Emran, and M. M. Richter, "A flexible method for software effort estimation by analogy," *Empirical Software Engineering*, vol. 12, no. 1, pp. 65–106, February, 2007.

[15] J. Li and G. Ruhe, "Software effort estimation by analogy using attribute selection based on rough set analysis," *International Journal of Software Engineering and Knowledge Engineering*, vol. 18, pp. 1–23, November 2008.

[16] B. T. Rao, B. Sameet, G. K. Swathi, K. V. Gupta, C. RaviTeja, and S. Sumana, "A novel neural network approach for software cost estimation using functional link artificial neural network," *International Journal of Computer Science and Network Security*, vol. 9, no. 6, pp. 126–131, June, 2009.

[17] Y. F. Li, M. Xie, and T. Goh, "A study of the non-linear adjustment for analogy based software cost estimation," *Empirical Software Engineering*, vol. 14, no. 6, pp. 603–643, February, 2009.

[18] P. Reddy, K. Sudha, P. R. Sree, and S. Ramesh, "Software effort estimation using radial basis and generalized regression neural networks," *arXiv preprint arXiv:1005.4021*, May, 2010.

[19] K. Dejaeger, W. Verbeke, D. Martens, and B. Baesens, "Data mining techniques for software effort estimation: a comparative study," *Software Engineering, IEEE Transactions on*, vol. 38, no. 2, pp. 375–397, April, 2012.

[20] E. Kocaguneli, T. Menzies, A. Bener, and J. W. Keung, "Exploiting the essential assumptions of analogy-based effort estimation," *Software Engineering, IEEE Transactions on*, vol. 38, no. 2, pp. 425–438, April, 2012.

[21] T. Menzies, Z. Chen, J. Hihn, and K. Lum, "Selecting best practices for effort estimation," *Software Engineering, IEEE Transactions on*, vol. 32, no. 11, pp. 883–895, November, 2006.

[22] J. Li and G. Ruhe, "Analysis of attribute weighting heuristics for analogy-based software effort estimation method aqua+," *Empirical Software Engineering*, vol. 13, no. 1, pp. 63–96, February, 2008.

[23] M. A. Hall and G. Holmes, "Benchmarking attribute selection techniques for discrete class data mining," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 15, no. 6, pp. 1437–1447, June, 2003.

[24] Z. Pawlak, "Rough sets," *International Journal of Computer & Information Sciences*, vol. 11, no. 5, pp. 341–356, September, 1982.

[25] B. Walczak and D. Massart, "Rough sets theory," *Chemometrics and intelligent laboratory systems*, vol. 47, no. 1, pp. 1–16, April, 1999.

[26] J. Dougherty, R. Kohavi, and M. Sahami, "Supervised and unsupervised discretization of continuous features," in *ICML*, pp. 194–202, 1995.

[27] L. C. Briand, V. R. Basili, and W. M. Thomas, "A pattern recognition approach for software engineering data analysis," *Software Engineering, IEEE Transactions on*, vol. 18, no. 11, pp. 931–942, November, 1992.

[28] S. Rajasekaran and G. A. Vijayalakshmi Pai, *Neural Networks, Fuzzy Logic And Genetic Algorithm: Synthesis And Applications*. PHI Learning Pvt.Ltd, 2003.

[29] Y. H. Pao, *Adaptive pattern recognition and neural networks*. Addison-Wesley Longman Publishing Co., Inc., 1989.

[30] D. W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *Journal of the Society for Industrial & Applied Mathematics*, vol. 11, no. 2, pp. 431–441, 1963.

[31] D. D. Lewis, "Naive (bayes) at forty: The independence assumption in information retrieval," in *Machine learning: ECML-98*, pp. 4–15, Springer, April 1998.

[32] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and regression trees*. CRC press, 1984.

[33] C. W. Hsu and C. J. Lin, "A comparison of methods for multiclass support vector machines," *Neural Networks, IEEE Transactions on*, vol. 13, no. 2, pp. 415–425, March, 2002.

[34] J. Han, M. Kamber, and J. Pei, *Data Mining, Second Edition: Concepts and Techniquesl*. Elsevier, 2006.

[35] R. Hecht-Nielsen, "Theory of the backpropagation neural network," in *Neural Networks, 1989. IJCNN., International Joint Conference on*, pp. 593–605, IEEE, 1989.

Dissemination of Work

Accepted

1. Jyoti Shivhare and Santanu Ku. Rath 'Software Effort Estimation using Machine Learning Techniques' *in Proceedings of ISEC: 7^{th} India Software Engineering Conference, IIT Madras, Feb 2014.*